```
RRRRRRRRRRR      MMM          MMM     SSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSSSSS
RRR         RRR  MMMMMM    MMMMMM   SSS
RRR         RRR  MMMMMM    MMMMMM   SSS
RRR         RRR  MMMMMM    MMMMMM   SSS
RRR         RRR  MMM  MMM  MMM  MMM  SSS
RRR         RRR  MMM  MMM  MMM  MMM  SSS
RRR         RRR  MMM  MMM  MMM  MMM  SSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSS
RRR   RRR        MMM          MMM             SSS
RRR   RRR        MMM          MMM             SSS
RRR   RRR        MMM          MMM             SSS
RRR      RRR     MMM          MMM             SSS
RRR      RRR     MMM          MMM             SSS
RRR      RRR     MMM          MMM             SSS
RRR         RRR  MMM          MMM  SSSSSSSSSSS
RRR         RRR  MMM          MMM  SSSSSSSSSSS
RRR         RRR  MMM          MMM  SSSSSSSSSSS
```

FILEID**RMOBUFMGR

```
RRRRRRRR    MM      MM    000000    BBBBBBBB    UU      UU  FFFFFFFFFF  MM      MM    GGGGGGGG  RRRRRRRR
RRRRRRRR    MM      MM    000000    BBBBBBBB    UU      UU  FFFFFFFFFF  MM      MM    GGGGGGGG  RRRRRRRR
RR      RR  MMMM  MMMM  00      00  BB      BB  UU      UU  FF          MMMM  MMMM  GG            RR      RR
RR      RR  MMMM  MMMM  00      00  BB      BB  UU      UU  FF          MMMM  MMMM  GG            RR      RR
RR      RR  MM  MM  MM  00    0000  BB      BB  UU      UU  FF          MM  MM  MM  GG            RR      RR
RR      RR  MM  MM  MM  00    0000  BB      BB  UU      UU  FF          MM  MM  MM  GG            RR      RR
RRRRRRRR    MM      MM  00  00  00  BBBBBBBB    UU      UU  FFFFFFFF    MM      MM  GG            RRRRRRRR
RRRRRRRR    MM      MM  00  00  00  BBBBBBBB    UU      UU  FFFFFFFF    MM      MM  GG            RRRRRRRR
RR  RR      MM      MM  0000    00  BB      BB  UU      UU  FF          MM      MM  GG  GGGGGG  RR  RR
RR  RR      MM      MM  0000    00  BB      BB  UU      UU  FF          MM      MM  GG  GGGGGG  RR  RR
RR    RR    MM      MM  00      00  BB      BB  UU      UU  FF          MM      MM  GG      GG  RR    RR      ....
RR    RR    MM      MM  00      00  BB      BB  UU      UU  FF          MM      MM  GG      GG  RR    RR      ....
RR      RR  MM      MM    000000    BBBBBBBB    UUUUUUUUUU  FF          MM      MM    GGGGGG    RR      RR    ....
RR      RR  MM      MM    000000    BBBBBBBB    UUUUUUUUUU  FF          MM      MM    GGGGGG    RR      RR    ....

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

Table of contents

```
0000     1          $BEGIN  RMOBUFMGR,000,RM$RMS0,<BUFFER MANAGER>
0000     2
0000     3 ;
0000     4 ;*****************************************************************************
0000     5 ;*                                                                           *
0000     6 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     7 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     8 ;*  ALL RIGHTS RESERVED.                                                     *
0000     9 ;*                                                                           *
0000    10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    11 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    13 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    15 ;*  TRANSFERRED.                                                             *
0000    16 ;*                                                                           *
0000    17 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    18 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    19 ;*  CORPORATION.                                                             *
0000    20 ;*                                                                           *
0000    21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    23 ;*                                                                           *
0000    24 ;*                                                                           *
0000    25 ;*****************************************************************************
0000    26 ;
```

```
0000     28 ;++
0000     29 ;
0000     30 ; Facility: rms32
0000     31 ;
0000     32 ; Abstract:
0000     33 ;                   this module contains the basic buffer management
0000     34 ;                   routines for rms32.  the following routines are
0000     35 ;                   included:
0000     36 ;
0000     37 ;                           rm$getpag       allocate empty pages
0000     38 ;                           rm$getspc       allocate space
0000     39 ;                           rm$retpag       deallocate pages
0000     40 ;                           rm$retspc       deallocate space
0000     41 ;                           rm$aldbuf       allocate bdb and i/o buffer
0000     42 ;                           rm$albdb        allocate bdb
0000     43 ;                           rm$alblb        allocate blb
0000     44 ;                           rm$retbdb       deallocate bdb and i/o
0000     45 ;                                           buffer (if any)
0000     46 ;
0000     47 ; Also included are routines and an entry-point to allocate and deallocate
0000     48 ; journaling BDB/Buffers.  These are:
0000     49 ;
0000     50 ;                           rm$aldjnlbuf    allocate journal BDB and buffer
0000     51 ;                           rm$aljnlbdb     allocate journal BDB
0000     52 ;                           rm$retjnlbdb    deallocate above
0000     53 ;
0000     54 ; Environment:
0000     55 ;                   star processor running starlet exec.
0000     56 ;
0000     57 ; Author: L F Laverdure , creation date: 30-DEC-1976
0000     58 ;
0000     59 ; Modified By:
0000     60 ;
0000     61 ;       V03-017 JWT0173         Jim Teague              1-Apr-1984
0000     62 ;               Disable new memory allocation for now.
0000     63 ;
0000     64 ;       V03-016 JWT0170         Jim Teague             22-Mar-1984
0000     65 ;               Improve memory deallocation in RM$RETSPC -- if we
0000     66 ;               can find enough consecutive scraps to make a page,
0000     67 ;               return the page then and there.
0000     68 ;
0000     69 ;       V03-015 RAS0263         Ron Schaefer            6-Mar-1984
0000     70 ;               Fine-tune things a little to improve the performance
0000     71 ;               a tad.
0000     72 ;
0000     73 ;       V03-014 RAS0219         Ron Schaefer            9-Dec-1983
0000     74 ;               Add RM$GETBLK1 entry point.
0000     75 ;
0000     76 ;       V03-012 KPL0004         Peter Lieberwirth       5-Aug-1983
0000     77 ;               Fix bug in V03-011.
0000     78 ;
0000     79 ;       V03-011 KPL0003         Peter Lieberwirth      27-Jul-1983
0000     80 ;               Add routines to allocate and deallocate journaling specific
0000     81 ;               buffers and BDBs.
0000     82 ;
0000     83 ;       V03-010 KPL0002         Peter Lieberwirth      30-Apr-1983
0000     84 ;               Add omitted macro definition.
```

```
0000     85 ;
0000     86 ;       V03-009 KPL0001         Peter Lieberwirth       29-Apr-1983
0000     87 ;               Add ASSUME for MJB to insure its longword aligned.
0000     88 ;
0000     89 ;       V03-009 MCN0001         Maria del C. Nasr       24-Mar-1983
0000     90 ;               Preserve register R2 to R4 in call to RM$ALBLB so that we
0000     91 ;               can use for this routine one of the general linkages defined.
0000     92 ;
0000     93 ;       V03-008 RAS0130         Ron Schaefer            14-Mar-1983
0000     94 ;               Change BDB allocation/deallocation to use the
0000     95 ;               new fields BDB$L_ALLOC_ADDR and BDB$W_ALLOC_SIZE.
0000     96 ;
0000     97 ;       V03-007 KBT0470         Keith B. Thompson       24-Jan-1983
0000     98 ;               Remove ret1stbdb hack
0000     99 ;
0000    100 ;       V03-006 KBT0453         Keith B. Thompson       6-Jan-1983
0000    101 ;               Put in assume statements to check the BLN to make
0000    102 ;               sure they are longword aligned
0000    103 ;
0000    104 ;       V03-005 RAS0106         Ron Schaefer            3-Dec-1982
0000    105 ;               Change the $SETPRT logic to only occur when memory is
0000    106 ;               actually created, not on every image I/O segment page.
0000    107 ;               The image activator has taken care of the image I/O segment.
0000    108 ;
0000    109 ;       V03-004 RAS0099         Ron Schaefer            22-Sep-1982
0000    110 ;               Change the $EXPREG logic to allocate a big block (128 pages)
0000    111 ;               at a time and insert on the free list; rather than
0000    112 ;               just grabbing a page or 2.   This helps prevent
0000    113 ;               fragmentation of P0 space.
0000    114 ;
0000    115 ;       V03-003 KBT0338         Keith B. Thompson       10-Sep-1982
0000    116 ;               Remove gets0spc and rets0spc routines and associated code
0000    117 ;
0000    118 ;       V03-002 KBT0199         Keith B. Thompson       23-Aug-1982
0000    119 ;               Reorganize psects
0000    120 ;
0000    121 ;       V03-001 KBT0121         Keith B. Thompson       7-Aug-1982
0000    122 ;               Remove $sifbdef, $sfsbdef and some commented out code
0000    123 ;
0000    124 ;       V02-026 CDS0003         C Saether       17-Jan-1982
0000    125 ;               Add RM$ALGBPB and RM$RETGBPB routines.
0000    126 ;
0000    127 ;       V02-025 CDS0002         C Saether       9-Nov-1981
0000    128 ;               Add and remove pages from s0 page list in
0000    129 ;               kernel mode for multi-proc.
0000    130 ;
0000    131 ;       V02-024 CDS0001         C Saether       21-Aug-1981
0000    132 ;               Add RM$ALBLB to allocate and initialize BLB's.
0000    133 ;               Add RM$RETBLB to deallocate BLB's.
0000    134 ;               Remove BCB allocation routine.
0000    135 ;
0000    136 ;       V02-023 SPR34112        C Saether       16-Jun-1981
0000    137 ;               Always allocate and deallocate in 16 byte units.
0000    138 ;               This solves the growing invisible hole problem.
0000    139 ;               It also solves the irab alignment problem.
0000    140 ;
0000    141 ;       V02-022 REFORMAT        C Saether       30-Jul-1980     22:25
```

```
0000   142 ;
0000   143 ;       V021    CDS0046         C Saether       27-Oct-1979     13:40
0000   144 ;               change aldbuf so buff size=0 just allocates bdb.
0000   145 ;               store requested, not allocated size in bdb as buffer size to
0000   146 ;               fix bug if ffff (hex) bytes are requested for buffer.
0000   147 ;               keep count of current pages in use and max used ever pages
0000   148 ;               in shared file database.  all shared pages now allocated
0000   149 ;               initially by rmsshare - don't allocate on demand.
0000   150 ;
0000   151 ;--
0000   152
```

```
                     .SBTTL  DECLARATIONS

;
; Include Files:
;

;
; Macros:
;

        $ASBDEF                 ; asb
        $BDBDEF                 ; bdb
        $BLBDEF                 ; blb
        $FWADEF                 ; fwa
        $GBDDEF                 ; gbd
        $GBHDEF                 ; gbh
        $GBPBDEF                ; gbpb
        $GBSBDEF                ; gbsb
        $IDXDEF                 ; idx
        $IFBDEF                 ; ifab
        $IRBDEF                 ; irab
        $RLBDEF                 ; rlb
        $RJBDEF                 ; rjb
        $SFSBDEF                ; sfsb
        $SLBDEF                 ; slb
        $SWBDEF                 ; swb
        $VADEF                  ; virtual address definitions
        $CMKRNLDEF              ; change mode to kernel argument defs
        $PTEDEF                 ; define page table entry defs
        $ACBDEF                 ; ast control block definitions
        $PSLDEF                 ; psl definitions
        $PCBDEF                 ; process control block definitions
        $IMPDEF                 ; impure area definitions
        $PRTDEF
        $IRPDEF
        $RMSDEF
        $MJBDEF                 ; miscellaneous journaling buffer definitions

;
; Equated Symbols:
;

;
; Own Storage:
;

000001FF 0000 C511:     .LONG   511                     ; constant for getting to page boundaries
000001FF 0004           MASK=   ^X1FF                   ; mask for getting to page boundary
```

```
0004   203          .SBTTL  RM$GETPAG - PAGE ALLOCATION ROUTINE
0004   204 ;++
0004   205 ;
0004   206 ; RM$GET1PAG - entry point to get only a single page
0004   207 ; RM$GETPAG - entry point to get requested number of pages
0004   208 ;
0004   209 ;  this routine allocates a specified number of pages.
0004   210 ;  the pages are not zero filled.
0004   211 ;
0004   212 ;  it performs this function by scanning the free page
0004   213 ;  list of the current (process or image) i/o segment
0004   214 ;  for the first fit.
0004   215 ;
0004   216 ;  if the request cannot be satisfied from the
0004   217 ;   free page list, the routine checks for new
0004   218 ;  (i.e. never-used) pages available in the i/o segment
0004   219 ;  and allocates from there.
0004   220 ;
0004   221 ;  if insufficient pages in the i/o segment then
0004   222 ;       if in the process i/o segment
0004   223 ;        or if rms is inhibited from using program
0004   224 ;        region (p0) space, return a dme error,
0004   225 ;       else allocate the page(s) from the program
0004   226 ;        region.
0004   227 ;
0004   228 ;  the pages will have exec write protection and
0004   229 ;  either supervisor or user read depending upon
0004   230 ;  the current i/o segment (i.e., for process and user
0004   231 ;  i/o segments respectively).
0004   232 ;
0004   233 ;
0004   234 ; Calling sequence:
0004   235 ;
0004   236 ;       BSBW    RM$GETPAG
0004   237 ;
0004   238 ;  alternate entry at rm$get1pag to allocate a single
0004   239 ;  page of memory.  same outputs but only r11 input required.
0004   240 ;
0004   241 ; Input Parameters:
0004   242 ;
0004   243 ;       r11     impure area pointer
0004   244 ;       r2      # of bytes required
0004   245 ;
0004   246 ; Implicit Inputs:
0004   247 ;
0004   248 ;       none
0004   249 ;
0004   250 ; Output Parameters:
0004   251 ;
0004   252 ;       r3      addr of starting page
0004   253 ;       r2      total length of buffer allocated
0004   254 ;               (i.e., r2 on input rounded up
0004   255 ;                to next page boundary)
0004   256 ;       r0      status code
0004   257 ;       r1      destroyed
0004   258 ;
0004   259 ; Implicit Outputs:
```

```
                     0004   260 ;
                     0004   261 ;       the affected free page list is updated.
                     0004   262 ;
                     0004   263 ; Completion Codes:
                     0004   264 ;
                     0004   265 ;       standard rms.  in particular, success or rm$_dme.
                     0004   266 ;
                     0004   267 ; Side Effects:
                     0004   268 ;
                     0004   269 ;       none
                     0004   270 ;
                     0004   271 ;--
                     0004   272
                     0004   273 RM$GET1PAG::
   52   01   D0      0004   274         MOVL    #1,R2                   ; asking for 1 byte gets 1 page
                     0007   275
                     0007   276 ;
                     0007   277 ;  setup to scan free page list for first fit
                     0007   278 ;
                     0007   279
                     0007   280 RM$GETPAG::
52   F6 AF   C0      0007   281         ADDL2   C511,R2                 ; round up required size
52   F2 AF   CA      000B   282         BICL2   C511,R2                 ; to length of pages
                     000F   283
                     000F   284 ;
                     000F   285 ;  scan free page list
                     000F   286 ;
                     000F   287
51   0C AB   DE      000F   288         MOVAL   IMP$L_FREEPGLH(R11),R1  ; get free page list head
10 AB   51   D1      0013   289 10$:    CMPL    R1,IMP$L_FREEPGLH+4(R11) ; end of list?
             1F   13 0017   290         BEQL    30$                     ; branch if yes
        51   61   D0 0019   291         MOVL    (R1),R1                 ; get next node
52   08 A1   F1      001C   292         CMPL    8(R1),R2                ; long enough?
             0D   19 0020   293         BLSS    10$                     ; branch if not
                  13 0022   294         BEQL    20$                     ; branch if exact fit
                     0024   295
                     0024   296 ;
                     0024   297 ;  we have a fit but we don't need all of the pages in the hole
                     0024   298 ;  - must return extras
                     0024   299 ;
                     0024   300
   50   52   51   C1 0024   301 15$:    ADDL3   R1,R2,R0                ; get addr of new hole
08 A0   08 A1   52   C3 0028   302         SUBL3   R2,8(R1),8(R0)          ; compute and store its length
             61   60   0E 002E   303         INSQUE  (R0),(R1)               ; insert the new hole
                     0031   304
                     0031   305 ;
                     0031   306 ;  the hole at r1 is just the right size
                     0031   307 ;  remove it from the list
                     0031   308 ;
                     0031   309
        53   61   0F 0031   310 20$:    REMQUE  (R1),R3                 ; address of hole to r3
                     0034   311 22$:    RMSSUC
                  05 0037   312         RSB
                     0038   313
                     0038   314
08 AB   52   D1      0038   315 30$:    CMPL    R2,IMP$L_IOSEGLEN(R11)  ; enough space?
             0E   14 003C   316         BGTR    200$                    ; branch if not
```

```
                     003E   317
                     003E   318 ;
                     003E   319 ;  take the required space from the i/o segment
                     003E   320 ;
                     003E   321
 53    04 AB    D0   003E   322          MOVL    IMP$L_IOSEGADDR(R11),R3 ; addr of space
 08 AB    52    C2   0042   323          SUBL2   R2,IMP$L_IOSEGLEN(R11)  ; adjust length of remaining
                     0046   324                                          ; space
 04 AB    52    C0   0046   325          ADDL2   R2,IMP$L_IOSEGADDR(R11) ; and its start addr
          E8    11   004A   326          BRB     22$                     ; and return success
                     004C   327
                     004C   328 ;
                     004C   329 ;  there is no space in the free page list or in the i/o segment.
                     004C   330 ;  if this is not the pio segment, allocate the required space
                     004C   331 ;  from the program region unless prohibited by user.
                     004C   332 ;
                     004C   333
                     004C   334          ASSUME  IMP$W_RMSSTATUS EQ      0
                     004C   335          ASSUME  IMP$V_IIOS      EQ      0
                     004C   336
       09 6B    E9   004C   337 200$:    BLBC    (R11),205$              ; branch if process i/o seg.
       04 AB    D5   004F   338          TSTL    IMP$L_IOSEGADDR(R11)    ; is there any image i/o seg.?
          04    13   0052   339          BEQL    205$                    ; branch if none (error)
 03 68    05    E1   0054   340          BBC     #IMP$V_NOPOBUFS,(R11),210$ ; branch if p0 off limits
        0091    31   0058   341 205$:    BRW     ERRDME
                     0058   342
                     0058   343 ;
                     0058   344 ;  expand the program region
                     0058   345 ;
                     0058   346
 51    52    F7 8F   78   0058   347 210$:    ASHL    #-9,R2,R1               ; convert to pages
 51  00000080 8F   D1   0060   348          CMPL    #128,R1                 ; use max (128, request)
          04    1B   0067   349          BLEQU   220$
    51    80 8F   9A   0069   350          MOVZBL  #128,R1
          7E    7C   006D   351 220$:    CLRQ    -(SP)                   ; temp array to receive results
       53    5E    D0   006F   352          MOVL    SP,R3                   ; and save its addr
                     0072   353          $EXPREG_S       PAGCNT=R1,-     ; num of pages
                     0072   354                          RETADR=(R3),-   ; start/end addr of space
                     0072   355                          ACMODE=#PSL$C_EXEC,- ; owner mode
                     0072   356                          REGION=#0       ; program region
       2E 50    E9   0081   357          BLBC    R0,EXPREGERR            ; got it!
                     0084   358
                     0084   359 ;
                     0084   360 ;  the required number of pages have now been allocated.
                     0084   361 ;  set the protection on them.
                     0084   362 ;
                     0084   363
    51    02 AB    9A   0084   364 SETPRT: MOVZBL  IMP$B_PROT(R11),R1      ; pick up protection for pages
                     0088   365          $SETPRT_S       INADR=(R3),-    ; start/end addr of space
                     0088   366                          RETADR=(R3),-   ; start/end addr of space
                     0088   367                          ACMODE=#PSL$C_EXEC,-
                     0088   368                          PROT=R1
       56 50    E9   0099   369          BLBC    R0,ERRBUG               ; service should not fail
       50    8E    D0   009C   370          MOVL    (SP)+,R0                ; addr of starting page
 51    8E    50    C3   009F   371          SUBL3   R0,(SP)+,R1             ; get length-1
             51    D6   00A3   372          INCL    R1                      ; and make it length
             34    BB   00A5   373          PUSHR   #^M<R2,R4,R5>           ; save regs
```

```
                54   50    7D  00A7   374          MOVQ    R0,R4                   ; copy start addr
                                00AA   375                                          ;  & length to proper regs
                   00DB    30  00AA   376          BSBW    RM$RETPAG               ; give pages to free list
                     34    BA  00AD   377          POPR    #^M<R2,R4,R5>           ; restore regs
                   FF55    31  00AF   378          BRW     RM$GETPAG               ; and go use the new space
                                00B2   379
                                00B2   380 EXPREGERR:
                     63    D5  00B2   381          TSTL    (R3)                    ; did we get anything?
                     34    19  00B4   382          BLSS    ERRDME1                 ; if not, report error
           50   04 A3    63    C3  00B6   383          SUBL3   (R3),4(R3),R0           ; how much did we get?
                     50    D6  00BB   384          INCL    R0                      ; correct length
              51   50    52    C3  00BD   385          SUBL3   R2,R0,R1                ; did we get enough?
                     22    19  00C1   386          BLSS    ERRDME2                 ; nope
                     BF    13  00C3   387          BEQL    SETPRT                  ; exact amount?
                04 A3    51    C2  00C5   388          SUBL2   R1,4(R3)                ; adjust addr array
                     02    10  00C9   389          BSBB    CNTREG                  ; give back unneeded pages
                     B7    11  00CB   390          BRB     SETPRT                  ; set the protection
                                00CD   391
           51   51   F7 8F    78  00CD   392 CNTREG: ASHL    #-9,R1,R1               ; convert to pages
                                00D2   393          $CNTREG_S       PAGCNT=R1,-     ; give back the excess space
                                00D2   394                          ACMODE=#PSL$C_EXEC,- ; owner mode
                                00D2   395                          REGION=#0       ; program region
                   0E 50    E9  00E1   396          BLBC    R0,ERRBUG               ; service should not fail
                               05  00E4   397          RSB
                                00E5   398
                                00E5   399 ;
                                00E5   400 ;  no dynamic memory available
                                00E5   401 ;
                                00E5   402 ERRDME2:
                51   50    D0  00E5   403          MOVL    R0,R1                   ; return all the space
                     E3    10  00E8   404          BSBB    CNTREG                  ; and report the error
                                00EA   405 ERRDME1:
                     03    BA  00EA   406          POPR    #^M<R0,R1>              ; clean up stack
                                00EC   407 ERRDME:
                                00EC   408          RMSERR  DME
                               05  00F1   409          RSB
                                00F2   410
                                00F2   411 ;
                                00F2   412 ;  the change protection system service failed
                                00F2   413 ;
                                00F2   414
                                00F2   415 ERRBUG: RMSTBUG FTL$_SETPRTFAIL
                                00F9   416
```

```
00F9   418          .SBTTL  RMSGETSPC - MEMORY ALLOCATION ROUTINE
00F9   419
00F9   420 ;++
00F9   421 ;
00F9   422 ; RMSGETSPC1 - set up free space header and get space
00F9   423 ; RMSGETSPC - get space
00F9   424 ; RMSGETSPC_ALT - yet another entry point to get space
00F9   425 ; RMSGETBLK - get space by longwords
00F9   426 ; RMSGETBLK1 - set up free space header and get space by longwords
00F9   427 ;
00F9   428 ;
00F9   429 ;  this routine allocates space within a page on a first
00F9   430 ;  fit basis.  the allocated space is zero filled.
00F9   431 ;
00F9   432 ;  if insufficient space is available, another page is
00F9   433 ;  added to the free space list.
00F9   434 ;
00F9   435 ;
00F9   436 ;  calling sequence:
00F9   437 ;
00F9   438 ;       BSBW    RMSGETSPC
00F9   439 ;
00F9   440 ;  alternate entry at rm$getspc_alt if r1 has exact address of list head
00F9   441 ;  alternate entry at rm$getblk if r2 has # of longwords required and
00F9   442 ;  this # is to be stored in byte 9 of the gotten space
00F9   443 ;
00F9   444 ;  input parameters:
00F9   445 ;
00F9   446 ;       r11     impure area addr
00F9   447 ;       r2      # of bytes required (11 < r2 < 513)
00F9   448 ;       r1      any address within page
00F9   449 ;               (space header must be at the start
00F9   450 ;                of this page)
00F9   451 ;
00F9   452 ;  implicit inputs:
00F9   453 ;
00F9   454 ;       the status of the impure area.
00F9   455 ;
00F9   456 ;  output parameters:
00F9   457 ;
00F9   458 ;       r1      addr of block of memory
00F9   459 ;       r0      status
00F9   460 ;       r2,r3,r4  destroyed
00F9   461 ;
00F9   462 ;  implicit outputs:
00F9   463 ;
00F9   464 ;  the free space list is updated.
00F9   465 ;
00F9   466 ;
00F9   467 ;  completion codes:
00F9   468 ;
00F9   469 ;       standard rms32, in particular, success and dme.
00F9   470 ;
00F9   471 ;  side effects:
00F9   472 ;
00F9   473 ;       none.
00F9   474 ;
```

```
                     00F9    475 ;--
                     00F9    476
                     00F9    477
                     00F9    478 ;
                     00F9    479 ;  alternate entry here for getting space from ifab free space list
                     00F9    480 ;
                     00F9    481 ;  additonal input:     r9 = address of ifab/irab
                     00F9    482 ;  r1 is not an input
                     00F9    483 ;
                     00F9    484
                     00F9    485 RMSGETSPC1::
          007E   30  00F9    486         BSBW    SETHDR1                 ; set up free space header page addr
                     00FC    487                                         ; and fall thru into rm$getspc
                     00FC    488 ;
                     00FC    489 ;  normal entry point
                     00FC    490 ;
                     00FC    491
                     00FC    492 RMSGETSPC::
    51    FF00 CF    CA  00FC    493         BICL2   C511,R1             ; get header addr
                     0101    494
                     0101    495 RMSGETSPC_ALT::
          54   51    D0  0101    496         MOVL    R1,R4               ; save addr for end test
          52   0F    C0  0104    497         ADDL2   #15,R2              ; turn request into multiple of
          52   0F    CA  0107    498         BICL2   #15,R2              ; 16 bytes.
                     010A    499
                     010A    500 ;
                     010A    501 ;  scan for first fit
                     010A    502 ;
                     010A    503
          54   61    D1  010A    504 10$:    CMPL    (R1),R4             ; end of list?
               32    13  010D    505         BEQL    50$                 ; branch if yes - no space found
          51   61    D0  010F    506 15$:    MOVL    (R1),R1             ; get next node
        52   08 A1    D1  0112    507         CMPL    8(R1),R2            ; long enough?
               F2    19  0116    508         BLSS    10$                 ; branch if not
               15    13  0118    509         BEQL    20$                 ; branch if exact fit
                     011A    510
                     011A    511 ;
                     011A    512 ;  we have a fit but don't need extra bytes
                     011A    513 ;  return them to the free space list
                     011A    514 ;
                     011A    515
     50   51    52    C1  011A    516         ADDL3   R2,R1,R0            ; get new hole addr
53   08 A1    52    C3  011E    517         SUBL3   R2,8(R1),R3         ; compute its length
          10   53    D1  0123    518         CMPL    R3,#16              ; at least 16 bytes?
               07    19  0126    519         BLSS    20$                 ; branch if not, as not
                     0128    520                                         ; big enough for a node
       08 A0   53    D0  0128    521         MOVL    R3,8(R0)            ; store hole length
          61   60    0E  012C    522         INSQUE  (R0),(R1)           ; & insert the new hole
                     012F    523
                     012F    524 ;
                     012F    525 ;  the hole at r1 is just the right size (imagine that!)
                     012F    526 ;  (actually could be 8 bytes longer than needed)
                     012F    527 ;  remove it from the list and zero fill it
                     012F    528 ;
                     012F    529
          51   61    0F  012F    530 20$:    REMQUE  (R1),R1
               55    DD  0132    531         PUSHL   R5
```

```
                61   52   00   61   00    2C  0134   532          MOVC5   #0,(R1),#0,R2,(R1)      ; zero the space, preserving R1
                                   55 8ED0  013A   533          POPL    R5
                                            013D   534          RMSSUC
                                       05   0140   535          RSB
                                            0141   536
                                            0141   537 ;
                                            0141   538 ;
                                            0141   539 ;  found no space of required size
                                            0141   540 ;  get another page and add it to the free space list
                                            0141   541 ;
                                            0141   542
                        7E   51   7D        0141   543 50$:    MOVQ    R1,-(SP)
                           FEC0   30        0144   544          BSBW    RMSGETPAG                   ; get required pages (r3 = addr)
                             A0   50   E9   0147   545          BLBC    R0,ERRDME1                  ; error if not available
                     08   A3      52   D0   014A   546          MOVL    R2,8(R3)                    ; store length of space
                           51      8E   7D   014E   547          MOVQ    (SP)+,R1
                                            0151   548
                                            0151   549 ;
                                            0151   550 ;  insert the new space in ascending memory address sequence
                                            0151   551 ;
                                            0151   552 ;  note: r1 points to last hole (i.e., highest in memory)
                                            0151   553 ;
                                            0151   554
                        51   53   D1        0151   555 60$:    CMPL    R3,R1                       ; is this the right spot?
                             09   1A        0154   556          BGTRU   70$                         ; branch if yes
                   51      04   A1   D0     0156   557          MOVL    4(R1),R1                    ; get previous hole
                        54   51   D1        015A   558          CMPL    R1,R4                       ; back at list head?
                             F2   12        015D   559          BNEQ    60$                         ; branch if not
                        61   63   0E        015F   560 70$:    INSQUE  (R3),(R1)                   ; insert the hole
                             AB   11        0162   561          BRB     15$                         ; and go use it
                                            0164   562
```

```
0164   564 ;
0164   565 ;       These assumes are to make sure that all of the structures
0164   566 ;       which are allocated with getblk are longword aligned.  If
0164   567 ;       they are not data in other structures can be corrupted.
0164   568 ;
0164   569 ;       asb
0164   570 ;
0164   571         ASSUME  <<ASB$K_BLN_FIX/4>*4>   EQ      ASB$K_BLN_FIX
0164   572         ASSUME  <<ASB$K_BLN_FAB/4>*4>   EQ      ASB$K_BLN_FAB
0164   573         ASSUME  <<ASB$K_BLN_SEQ/4>*4>   EQ      ASB$K_BLN_SEQ
0164   574         ASSUME  <<ASB$K_BLN_REL/4>*4>   EQ      ASB$K_BLN_REL
0164   575         ASSUME  <<ASB$K_BLN_IDX/4>*4>   EQ      ASB$K_BLN_IDX
0164   576 ;
0164   577 ;       bdb
0164   578 ;
0164   579         ASSUME  <<BDB$K_BLN/4>*4>       EQ      BDB$K_BLN
0164   580 ;
0164   581 ;       blb
0164   582 ;
0164   583         ASSUME  <<BLB$K_BLN/4>*4>       EQ      BLB$K_BLN
0164   584 ;
0164   585 ;       fwa  (the fwa is not allocated with getblk but may someday)
0164   586 ;
0164   587         ASSUME  <<FWA$K_BLN/4>*4>       EQ      FWA$K_BLN
0164   588         ASSUME  <<FWA$K_BLN_FWA/4>*4>   EQ      FWA$K_BLN_FWA
0164   589         ASSUME  <<FWA$K_BLN_BUF/4>*4>   EQ      FWA$K_BLN_BUF
0164   590 ;
0164   591 ;       gbd
0164   592 ;
0164   593         ASSUME  <<GBD$K_BLN/4>*4>       EQ      GBD$K_BLN
0164   594 ;
0164   595 ;       gbh
0164   596 ;
0164   597         ASSUME  <<GBH$K_BLN/4>*4>       EQ      GBH$K_BLN
0164   598 ;
0164   599 ;       gbpb
0164   600 ;
0164   601         ASSUME  <<GBPB$K_BLN/4>*4>      EQ      GBPB$K_BLN
0164   602 ;
0164   603 ;       gbsb
0164   604 ;
0164   605         ASSUME  <<GBSB$K_BLN/4>*4>      EQ      GBSB$K_BLN
0164   606 ;
0164   607 ;       idx
0164   608 ;
0164   609         ASSUME  <<IDX$K_FIXED_BLN/4>*4> EQ      IDX$K_FIXED_BLN
0164   610 ;
0164   611 ;       ifab
0164   612 ;
0164   613         ASSUME  <<IFB$K_BLN_SEQ/4>*4>   EQ      IFB$K_BLN_SEQ
0164   614         ASSUME  <<IFB$K_BLN_REL/4>*4>   EQ      IFB$K_BLN_REL
0164   615         ASSUME  <<IFB$K_BLN_IDX/4>*4>   EQ      IFB$K_BLN_IDX
0164   616 ;
0164   617 ;       irab
0164   618 ;
0164   619         ASSUME  <<IRB$K_BLN_SEQ/4>*4>   EQ      IRB$K_BLN_SEQ
0164   620         ASSUME  <<IRB$K_BLN_REL/4>*4>   EQ      IRB$K_BLN_REL
```

```
0164   621          ASSUME  <<IRB$K_BLN_IDX/4>*4>   EQ      IRB$K_BLN_IDX
0164   622 ;
0164   623 ;       mjb
0164   624 ;
0164   625          ASSUME  <<MJB$K_BLN/4>*4>       EQ      MJB$K_BLN
0164   626 ;
0164   627 ;       rlb
0164   628 ;
0164   629          ASSUME  <<RLB$K_BLN/4>*4>       EQ      RLB$K_BLN
0164   630 ;
0164   631 ;       rjb
0164   632 ;
0164   633          ASSUME  <<RJB$K_BLN/4>*4>       EQ      RJB$K_BLN
0164   634 ;
0164   635 ;       sfsb
0164   636 ;
0164   637          ASSUME  <<SFSB$K_BLN/4>*4>      EQ      SFSB$K_BLN
0164   638 ;
0164   639 ;       slb
0164   640 ;
0164   641          ASSUME  <<SLB$K_BLN/4>*4>       EQ      SLB$K_BLN
0164   642 ;
0164   643 ;       swb
0164   644 ;
0164   645          ASSUME  <<SWB$K_BLN/4>*4>       EQ      SWB$K_BLN
0164   646
```

```
                     0164   648 ;
                     0164   649 ;  alternate entry here for getting block from ifab free space list
                     0164   650 ;
                     0164   651 ;  additonal input:     r9 = address of ifab/irab
                     0164   652 ;  r1 is not an input
                     0164   653 ;
                     0164   654
                     0164   655 RMSGETBLK1::
              14  10   0164   656          BSBB    SETHDR1                         ; set up free space header page addr
                     0166   657                                                  ; and fall thru into rm$getblk
                     0166   658
                     0166   659 ;
                     0166   660 ;
                     0166   661 ;  alternate entry to get space by # of longwords and store that
                     0166   662 ;  number in byte 9 of the returned space
                     0166   663 ;
                     0166   664
                     0166   665 RMSGETBLK::
              52  DD   0166   666          PUSHL   R2                              ; save # longwords
        52    52  02   0168   667          ROTL    #2,R2,R2                        ; make into # bytes
            FF8D  30   016C   668          BSBW    RMSGETSPC                       ; go get the space
          05 50   E9   016F   669          BLBC    R0,10$                          ; get out on error
       09 A1      8E   F6   0172   670          CVTLB   (SP)+,9(R1)                     ; store length
                     0176   671          RSB
              02  BA   0177   672 10$:     POPR    #^M<R1>                         ; clean stack
                  05   0179   673          RSB
                     017A   674
                     017A   675 ;++
                     017A   676 ;  subroutine to load ifab addr into r1
                     017A   677 ;
                     017A   678 ;--
                     017A   679
                     017A   680 SETHDR1:
           51    59  D0   017A   681          MOVL    R9,R1                           ; assume ifab addr in r1
                     017D   682
                     017D   683          ASSUME  <IRB$C_BID&1>   EQ      0
                     017D   684          ASSUME  <IFB$C_BID&1>   EQ      1
                     017D   685          ASSUME  IFB$B_BID       EQ      IRB$B_BID
                     017D   686
     03 08 A9      E8   017D   687          BLBS    IFB$B_BID(R9),10$               ; branch if structure is ifab
           51    69  D0   0181   688          MOVL    IRB$L_IFAB_LNK(R9),R1           ; get ifab address from irab
                  05   0184   689 10$:     RSB
```

```
                     0185   691          .SBTTL  RMSRETPAG - PAGE DEALLOCATION ROUTINE
                     0185   692
                     0185   693 ;++
                     0185   694 ;
                     0185   695 ; RMSRET1PAG - return one (1) page
                     0185   696 ; RMSRETPAG - deallocate pages
                     0185   697 ;
                     0185   698 ;       this routine returns pages to the free page list.
                     0185   699 ;       the list is kept in order of ascending memory addresses.
                     0185   700 ;
                     0185   701 ;
                     0185   702 ;  calling sequence:
                     0185   703 ;
                     0185   704 ;       BSBW    RMSRETPAG
                     0185   705 ;
                     0185   706 ;  alternate entry at rm$ret1pag to return a single page.
                     0185   707 ;  r11,r4 are only inputs.
                     0185   708 ;
                     0185   709 ;  input parameters:
                     0185   710 ;
                     0185   711 ;       r11     impure area address
                     0185   712 ;       r5      length in bytes of pages to be returned
                     0185   713 ;       r4      address of first page to be returned
                     0185   714 ;
                     0185   715 ;  outputs:
                     0185   716 ;
                     0185   717 ;       r0 thru r5 destroyed
                     0185   718 ;
                     0185   719 ;  implicit outputs:
                     0185   720 ;
                     0185   721 ;       the free page list is updated.
                     0185   722 ;
                     0185   723 ;  completion codes:
                     0185   724 ;
                     0185   725 ;       none
                     0185   726 ;
                     0185   727 ;  side effects:
                     0185   728 ;
                     0185   729 ;       none
                     0185   730 ;
                     0185   731 ;--
                     0185   732 ;
                     0185   733
                     0185   734 ;
                     0185   735 ;  entry to return a single page
                     0185   736 ;
                     0185   737
                     0185   738 RMSRET1PAG::
     55   01   D0   0185   739          MOVL    #1,R5                   ; 1 byte gets 1 page
                     0188   740
                     0188   741 RMSRETPAG::
55   FE74 CF   C0   0188   742          ADDL2   C511,R5                 ; round up length
55   FE6F CF   CA   018D   743          BICL2   C511,R5                 ;  ''
54   FE6A CF   CA   0192   744          BICL2   C511,R4                 ; get start of page
  52   0C AB   DE   0197   745          MOVAL   IMPSL_FREEPGLH(R11),R2  ; addr of header
     53   52   D0   019B   746          MOVL    R2,R3                   ; save for end of list test
                     019E   747
```

```
                     019E   748 ;
                     019E   749 ;  scan for a hole having a higher address
                     019E   750 ;
                     019E   751
          52   62    D0   019E   752 10$:    MOVL    (R2),R2                 ; get next hole addr
          53   52    D1   01A1   753         CMPL    R2,R3                   ; end of list?
               18    13   01A4   754         BEQL    40$                     ; branch if yes
          54   52    D1   01A6   755         CMPL    R2,R4                   ; higher than hole being returned?
               F3    1F   01A9   756         BLSSU   10$                     ; branch if not
                          01AB   757
                          01AB   758 ;
                          01AB   759 ;  the hole at r2 has a higher address than that being returned.
                          01AB   760 ;  insert the returning hole and try to combine it with either the
                          01AB   761 ;  previous hole, the next hole, or both
                          01AB   762 ;
                          01AB   763
     51   54   55    C1   01AB   764 20$:    ADDL3   R5,R4,R1                ; get address past returning hole
          52   51    D1   01AF   765         CMPL    R1,R2                   ; same as start of next hole?
               0A    12   01B2   766         BNEQ    40$                     ; branch if not
                          01B4   767
                          01B4   768 ;
                          01B4   769 ;  combine this hole with next hole
                          01B4   770 ;
                          01B4   771
     55     08 A2    C0   01B4   772         ADDL2   8(R2),R5                ; get new hole size
          52   62    0F   01B8   773         REMQUE  (R2),R2                 ; get rid of high hole
          52   62    D0   01B8   774         MOVL    (R2),R2                 ; get next node addr
                          01BE   775                                         ; *****
                          01BE   776                                         ; NOTE:
                          01BE   777                                         ;       assumes the address
                          01BE   778                                         ;       in the deleted node is still
                          01BE   779                                         ;       valid! (it should be.)
                          01BE   780                                         ; ******
                          01BE   781
                          01BE   782 ;
                          01BE   783 ;  check if the hole can be combined with the previous hole
                          01BE   784 ;
                          01BE   785
     50     04 A2    D0   01BE   786 40$:    MOVL    4(R2),R0                ; get previous hole addr
          53   50    D1   01C2   787         CMPL    R0,R3                   ; is it the head?
               0F    13   01C5   788         BEQL    60$                     ; branch if yes
51   50     08 A0    C1   01C7   789         ADDL3   8(R0),R0,R1             ; get end of previous hole
          54   51    D1   01CC   790         CMPL    R1,R4                   ; same as start of hole
                          01CF   791                                         ; being returned?
               05    12   01CF   792         BNEQ    60$                     ; branch if not
     08 A0     55    C0   01D1   793         ADDL2   R5,8(R0)                ; just add in the additional
                          01D5   794                                         ; size and that's all
                     05   01D5   795         RSB                             ; return to caller
                          01D6   796
                          01D6   797 ;
                          01D6   798 ;  must create a new node for hole being returned
                          01D6   799 ;
                          01D6   800
     08 A4     55    D0   01D6   801 60$:    MOVL    R5,8(R4)                ; set its size
          60   64    0E   01DA   802         INSQUE  (R4),(R0)               ; and insert it
                     05   01DD   803 65$:    RSB
```

```
01DE    805          .SBTTL  RMSRETSPC - MEMORY DEALLOCATION ROUTINE
01DE    806
01DE    807 ;++
01DE    808 ;
01DE    809 ; RMSRETSPC1 - set up free list header and return space
01DE    810 ; RMSRETSPC - return space
01DE    811 ; RMSRETBLK1 - set up header and return space with length field
01DE    812 ; RMSRETBLK - return space with length field
01DE    813 ;
01DE    814 ;  this routine returns memory to the free space list, or to the system
01DE    815 ;  paged pool.
01DE    816 ;
01DE    817 ;  the list is kept in ascending memory sequence to facilitate
01DE    818 ;  combining holes.  holes are not combined across page
01DE    819 ;  boundaries however.
01DE    820 ;
01DE    821 ;  note that any hole less than 12 bytes in length is implicit
01DE    822 ;  (i.e., it has no header linking it into the list).
01DE    823 ;
01DE    824 ;
01DE    825 ;  calling sequence:
01DE    826 ;
01DE    827 ;       BSBW    RMSRETSPC
01DE    828 ;
01DE    829 ;  alternate entry at rm$retblk to return a block having
01DE    830 ;  its length stored as a # of longwords in byte 9 of the returning space.
01DE    831 ;  for this entry the r2 input is not required.
01DE    832 ;
01DE    833 ;  input parameters:
01DE    834 ;
01DE    835 ;       r11     impure area address
01DE    836 ;       r4      addr of space being returned
01DE    837 ;       r3      any addr in page having free space header
01DE    838 ;       r2      length in bytes of space being returned
01DE    839 ;
01DE    840 ;  inplicit inputs:
01DE    841 ;
01DE    842 ;       none
01DE    843 ;
01DE    844 ;  output parameters:
01DE    845 ;
01DE    846 ;       r0 thru r5 destroyed
01DE    847 ;
01DE    848 ;  implicit outputs:
01DE    849 ;
01DE    850 ;       the free space list is updated.
01DE    851 ;
01DE    852 ;  completion codes:
01DE    853 ;
01DE    854 ;       none.
01DE    855 ;
01DE    856 ;  side effects:
01DE    857 ;
01DE    858 ;       none
01DE    859 ;
01DE    860 ;--
01DE    861 ;
```

```
                     01DE    862
00000200             01DE    863 C512:   .LONG   512                     ; constant for page size
                     01E2    864
                     01E2    865 ;
                     01E2    866 ;  alternate entry to return space to ifab free space list
                     01E2    867 ;  additional input:    r9 = ifab/irab addr
                     01E2    868 ;  r3 is not an input
                     01E2    869 ;
                     01E2    870
                     01E2    871 RMS$RETSPC1::
                     01E2    872                                             ; get ifab addr into r3
      FA'AF    9F    01E2    873         PUSHAB  B^RMS$RETSPC                ; and go do normal return
                     01E5    874
                     01E5    875 ;++
                     01E5    876 ;  subroutine to load ifab addr into r3
                     01E5    877 ;
                     01E5    878 ;--
                     01E5    879
                     01E5    880 SETHDR3:
   53    59    D0    01E5    881         MOVL    R9,R3                       ; assume ifab addr in r3
                     01E8    882
                     01E8    883         ASSUME  <IRB$C_BID&1>   EQ      0
                     01E8    884         ASSUME  <IFB$C_BID&1>   EQ      1
                     01E8    885         ASSUME  IFB$B_BID       EQ      IRB$B_BID
                     01E8    886
03 08 A9    E8    01E8    887         BLBS    IFB$B_BID(R9),10$           ; branch if structure is ifab
 53    69    D0    01EC    888         MOVL    IRB$L_IFAB_LNK(R9),R3       ; get ifab address from irab
          05    01EF    889 10$:    RSB
                     01F0    890
                     01F0    891 ;
                     01F0    892 ;  alternate entry to return block to ifab free space list
                     01F0    893 ;  additional input:    r9 = ifab/irab addr
                     01F0    894 ;  r3 is not an input
                     01F0    895 ;
                     01F0    896
                     01F0    897 RMS$RETBLK1::
         F3    10    01F0    898         BSBB    SETHDR3                     ; get ifab addr into r3
                     01F2    899                                             ; and fall into normal return blk
                     01F2    900
                     01F2    901 ;
                     01F2    902 ;  alternate entry here to return a block having a length code
                     01F2    903 ;
                     01F2    904
                     01F2    905 RMS$RETBLK::
   52    09 A4    9A    01F2    906         MOVZBL  9(R4),R2                ; pick up length from block
52   52    02    9C    01F6    907         ROTL    #2,R2,R2                ; convert to # bytes
                     01FA    908
                     01FA    909 RMS$RETSPC::
      52    0F    C0    01FA    910         ADDL2   #15,R2                  ; always round to multiple
      52    0F    CA    01FD    911         BICL2   #15,R2                  ; of 16 bytes.
   DA AF    52    D1    0200    912 1$:     CMPL    R2,C512                 ; returning at least a page?
          17    19    0204    913         BLSS    5$                      ; branch if not
          54    DD    0206    914 3$:     PUSHL   R4                      ; save registers
      7E    52    7D    0208    915         MOVQ    R2,-(SP)
        FF77    30    020B    916         BSBW    RMS$RET1PAG             ; return 1 page to free page list
      52    8E    7D    020E    917         MOVQ    (SP)+,R2                ; restore registers
54    8E    CA AF    C1    0211    918         ADDL3   C512,(SP)+,R4           ; adjust address of returning space
```

```
                52   C5 AF    C2  0216   919          SUBL2   C512,R2                 ; addjust length of space left
                        E4    12  021A   920          BNEQ    1$                      ; branch if more space to return
                              05  021C   921          RSB
                53 FDDF CF    CA  021D   922 5$:       BICL2   C511,R3                 ; get free space list head addr
                     55 53    D0  0222   923          MOVL    R3,R5                   ; save for end test
                                  0225   924
                                  0225   925 ;
                                  0225   926 ;  scan for a hole having a higher address
                                  0225   927 ;
                                  0225   928
                     55 63    D1  0225   929 10$:      CMPL    (R3),R5                 ; end of list?
                        2A    13  0228   930          BEQL    50$                     ; branch if yes
                     53 63    D0  022A   931          MOVL    (R3),R3                 ; get next hole addr
                     54 53    D1  022D   932          CMPL    R3,R4                   ; higher than hole being returned?
                        F3    1F  0230   933          BLSSU   10$                     ; branch if not
                                  0232   934
                                  0232   935 ;
                                  0232   936 ;  the hole at r3 has a higher address than hole being returned.
                                  0232   937 ;  try to combine with either the next or the previous hole or both.
                                  0232   938 ;
                                  0232   939
                                  0232   940 20$:                                   ; both buffers in same page?
                51 54 53    CD    0232   941          XORL3   R3,R4,R1                ; set bits 9-31 to 0
                                  0236   942                                        ; if in same page
                51 FDC6 CF    CA  0236   943          BICL2   C511,R1                 ; clear bits 0-8
                                  023B   944                                        ; z - set if 9-31 also zero
                        13    12  023B   945          BNEQ    40$                     ; branch if not
                51 53 54    C3  023D   946          SUBL3   R4,R3,R1                ; get difference between the
                                  0241   947                                        ; buffer addresses
                     51 52    C2  0241   948          SUBL2   R2,R1                   ; less the length of the
                                  0244   949                                        ; returning buffer
                        0A    12  0244   950          BNEQ    40$                     ; branch if not exact
                                  0246   951
                                  0246   952 ;
                                  0246   953 ;  combine the returning hole with the next hole
                                  0246   954 ;
                                  0246   955
                52   08 A3    C0  0246   956          ADDL2   8(R3),R2                ; get new hole size
                                  024A   957                                        ; = old + new
                   53   63    0F  024A   958          REMQUE  (R3),R3                 ; get rid of high hole
                   53   63    D0  024D   959          MOVL    (R3),R3                 ; get addr of next higher
                                  0250   960                                        ; hole
                                  0250   961                                        ;NOTE:
                                  0250   962                                        ; note: assumes link still
                                  0250   963                                        ;  valid.
                                  0250   964
                                  0250   965 ;
                                  0250   966 ;  check if hole can be combined with previous hole
                                  0250   967 ;
                                  0250   968
                53   04 A3    D0  0250   969 40$:      MOVL    4(R3),R3                ; get addr previous hole
                     55 53    D1  0254   970 50$:      CMPL    R3,R5                   ; is it the head?
                        1E    13  0257   971          BEQL    60$                     ; branch if yes
                                  0259   972                                        ; two buffers in same page?
              51 54 53    CD    0259   973          XORL3   R3,R4,R1                ; set bits 9-31 to 0
                                  025D   974                                        ; if in same page
              51 FD9F CF    CA  025D   975          BICL2   C511,R1                 ; clear bits 0-8
```

```
                 0262   976                                    ; Z - set if 9-31 also zero
            13   12   0262   977          BNEQ    60$            ; branch if not
      51    54   53   C3   0264   978          SUBL3   R3,R4,R1       ; get difference between the
                          0268   979                                 ; buffer addresses
        51    08 A3   C2   0268   980          SUBL2   8(R3),R1       ; less previous buffer length
            09   12   026C   981          BNEQ    60$            ; branch if not exact
        08 A3   52   C0   026E   982          ADDL2   R2,8(R3)       ; merely add size of new returning
                          0272   983                                 ; space to previous hole size
           54   53   D0   0272   984          MOVL    R3,R4          ; copy hole addr
            07   11   0275   985          BRB     80$            ; go check end condition
                          0277   986 ;
                          0277   987 ;  create a new node for hole being returned
                          0277   988 ;
        08 A4   52   D0   0277   989 60$:    MOVL    R2,8(R4)       ; set hole size
           63   64   0E   027B   990          INSQUE  (R4),(R3)      ; & insert it
                          027E   991
                          027E   992 ;
                          027E   993 ;  check for invisible space at end of page and reclaim if any
                          027E   994 ;
                          027E   995
      51    54    08 A4   C1   027E   996 80$:    ADDL3   8(R4),R4,R1    ; addr + size
    51   FFFFFE00 8F   CA   0283   997          BICL2   #^C MASK,R1    ; get offset in page
        51    01F8 8F   A2   028A   998          SUBW2   #512-8,R1      ; 8 bytes from end?
            04   12   028F   999          BNEQU   90$            ; branch if not
        08 A4    08   A0   0291  1000          ADDW2   #8,8(R4)       ; update the length
                          0295  1001
                          0295  1002 ;
                          0295  1003 ; If we have accumulated a page from all these scraps, we can give it back
                          0295  1004 ; ***
                          0295  1005 ;90$:    CMPL    C512,R2        ; did we manage to scrape up a page?
                          0295  1006 ;        BNEQ    95$            ; if not, continue
                          0295  1007 ;        REMQUE  (R4),R4        ; if so, remove from free list
                          0295  1008 ;        BSBW    RM$RET1PAG     ; and give back this page now
                          0295  1009 ; ***
                          0295  1010 90$:
                    05    0295  1011 95$:    RSB
                          0296  1012
```

```
                     0296  1014          .SBTTL  RMSALDBUF - BDB AND I/O BUFFER ALLOCATION ROUTINE
                     0296  1015
                     0296  1016 ;++
                     0296  1017 ;
                     0296  1018 ; RMSALDBUF - allocate buffer and bdb to go with it
                     0296  1019 ;
                     0296  1020 ;  this routine performs the following functions:
                     0296  1021 ;
                     0296  1022 ;       1. allocate and init a bdb
                     0296  1023 ;       2. allocate an i/o buffer (of an integral
                     0296  1024 ;          number of pages) if r5 non-zero
                     0296  1025 ;
                     0296  1026 ;  calling sequence:
                     0296  1027 ;
                     0296  1028 ;       BSBW    RMSALDBUF
                     0296  1029 ;
                     0296  1030 ;  input parameters:
                     0296  1031 ;
                     0296  1032 ;       r11     impure area address
                     0296  1033 ;       r10     ifab address
                     0296  1034 ;       r5      length of buffer in bytes
                     0296  1035 ;
                     0296  1036 ;  implicit inputs:
                     0296  1037 ;
                     0296  1038 ;       none
                     0296  1039 ;
                     0296  1040 ;  output parameters:
                     0296  1041 ;
                     0296  1042 ;       r4      address of bdb
                     0296  1043 ;       r3      address of buffer
                     0296  1044 ;       r2      total size in bytes of allocation
                     0296  1045 ;       r0      status code
                     0296  1046 ;       r1      destroyed
                     0296  1047 ;       r4,r5   also destroyed if r0 indicates an error
                     0296  1048 ;
                     0296  1049 ;  implicit outputs:
                     0296  1050 ;
                     0296  1051 ;  the affected free space and free page lists are updated.
                     0296  1052 ;
                     0296  1053 ;  completion codes:
                     0296  1054 ;
                     0296  1055 ;       standard rms, in particular, success or dme.
                     0296  1056 ;
                     0296  1057 ;  side effects:
                     0296  1058 ;
                     0296  1059 ;       none.
                     0296  1060 ;
                     0296  1061 ;--
                     0296  1062
                     0296  1063 RMSALDBUF::
          27   10    0296  1064          BSBB    RMSALBDB                ; get a bdb
        1E 50   E9    0298  1065          BLBC    R0,20$                  ; branch on error
    54  51   D0    029B  1066          MOVL    R1,R4                   ; save bdb addr
    52  55   D0    029E  1067          MOVL    R5,R2                   ; move buffer len to right reg
          16   13    02A1  1068          BEQL    20$                     ; eql then wants only bdb (no buffer)
       FD61   30    02A3  1069          BSBW    RMSGETPAG               ; and get an i/o buffer
                    02A6  1070                                          ; (len/addr returned in r2,r3)
```

```
                11 50   E9  02A6  1071          BLBC    R0,50$                  ; branch on error
                            02A9  1072
                            02A9  1073 ;
                            02A9  1074 ;  set buffer size and address into bdb
                            02A9  1075 ;
                            02A9  1076
          16 A4   55    B0  02A9  1077          MOVW    R5,BDB$W_SIZE(R4)
          2C A4   55    B0  02AD  1078          MOVW    R5,BDB$W_ALLOC_SIZE(R4)
          18 A4   53    D0  02B1  1079          MOVL    R3,BDB$L_ADDR(R4)
          28 A4   53    D0  02B5  1080          MOVL    R3,BDB$L_ALLOC_ADDR(R4)
                        05  02B9  1081 20$:     RSB                             ; note: r0 still valid.
                            02BA  1082
                            02BA  1083 ;
                            02BA  1084 ;  error allocating the page  -  must return the bdb
                            02BA  1085 ;
                            02BA  1086
                      4B    10  02BA  1087 50$:     BSBB    RM$RETBDB
                            02BC  1088
                            02BC  1089 ERRDME_BR:
                    FE2D    31  02BC  1090          BRW     ERRDME                  ; restore error code
```

```
                     1092          .SBTTL  RM$ALBDB - BDB ALLOCATION ROUTINE
                     1093
                     1094 ;++
                     1095 ;
                     1096 ; RM$ALBDB -  allocate and initialize a buffer descriptor block (bdb)
                     1097 ;
                     1098 ;  calling sequence:
                     1099 ;
                     1100 ;       BSBW    RM$ALBDB
                     1101 ;
                     1102 ;  input parameters:
                     1103 ;
                     1104 ;       r11     impure area address
                     1105 ;       r10     ifab address
                     1106 ;
                     1107 ;  implicit inputs:
                     1108 ;
                     1109 ;       none
                     1110 ;
                     1111 ;  output parameters:
                     1112 ;
                     1113 ;       r1      address of bdb
                     1114 ;       r0      status code
                     1115 ;       r2,r3,r4 destroyed
                     1116 ;
                     1117 ;  implicit outputs:
                     1118 ;
                     1119 ;  the bdb has its block length and block id fields filled in
                     1120 ;  and it is linked into the ifab's bdb list.
                     1121 ;
                     1122 ;  completion codes:
                     1123 ;
                     1124 ;       standard rms, in particular, success and dme.
                     1125 ;
                     1126 ;  side effects:
                     1127 ;
                     1128 ;       none
                     1129 ;
                     1130 ;--
                     1131
                     1132 RM$ALBDB::
     51   5A   D0    1133          MOVL    R10,R1                  ; copy ifab addr as this
                      1134                                          ; defines the page for the
                      1135                                          ; free space list header
     52   14   D0    1136          MOVL    #BDB$C_BLN/4,R2         ; # longwords required
        FE9E   30    1137          BSBW    RM$GETBLK               ; allocate zeroed space
                      1138                                          ; (r1 set to addr)
      08 50   E9     1139          BLBC    R0,10$                  ; branch on error
                      1140
                      1141 ;
                      1142 ;  set id into bdb and link at end of the ifab's bdb list
                      1143 ;
                      1144
08 A1   0C   90       1145          MOVB    #BDB$C_BID,BDB$B_BID(R1)
44 BA   61   0E       1146          INSQUE  (R1),@IFB$L_BDB_BLNK(R10)
              05      1147 10$:     RSB                             ; note: r0 still has status code.
```

```
                     02D4  1149          .SBTTL  RM$ALGBPB - GBPB ALLOCATION ROUTINE
                     02D4  1150
                     02D4  1151 ;++
                     02D4  1152 ;
                     02D4  1153 ; RM$ALGBPB -  allocate and initialize a buffer descriptor block (gbpb)
                     02D4  1154 ;
                     02D4  1155 ;  calling sequence:
                     02D4  1156 ;
                     02D4  1157 ;       BSBW    RM$ALGBPB
                     02D4  1158 ;
                     02D4  1159 ;  input parameters:
                     02D4  1160 ;
                     02D4  1161 ;       r11     impure area address
                     02D4  1162 ;       r10     ifab address
                     02D4  1163 ;
                     02D4  1164 ;  implicit inputs:
                     02D4  1165 ;
                     02D4  1166 ;       none
                     02D4  1167 ;
                     02D4  1168 ;  output parameters:
                     02D4  1169 ;
                     02D4  1170 ;       r1      address of gbpb
                     02D4  1171 ;       r0      status code
                     02D4  1172 ;       r2,r3,r4 destroyed
                     02D4  1173 ;
                     02D4  1174 ;  implicit outputs:
                     02D4  1175 ;
                     02D4  1176 ;  the gbpb has its block length and block id fields filled in
                     02D4  1177 ;  and it is linked into the ifab's gbpb list.
                     02D4  1178 ;
                     02D4  1179 ;  completion codes:
                     02D4  1180 ;
                     02D4  1181 ;       standard rms, in particular, success and dme.
                     02D4  1182 ;
                     02D4  1183 ;  side effects:
                     02D4  1184 ;
                     02D4  1185 ;       none
                     02D4  1186 ;
                     02D4  1187 ;--
                     02D4  1188
                     02D4  1189 RM$ALGBPB::
 51   5A    D0       02D4  1190          MOVL    R10,R1                  ; copy ifab addr as this
                     02D7  1191                                          ; defines the page for the
                     02D7  1192                                          ; free space list header
 52   0A    D0       02D7  1193          MOVL    #GBPB$C_BLN/4,R2                ; # longwords required
    FE89    30       02DA  1194          BSBW    RM$GETBLK               ; allocate zeroed space
                     02DD  1195                                          ; (r1 set to addr)
  08 50     E9       02DD  1196          BLBC    R0,10$                  ; branch on error
                     02E0  1197
                     02E0  1198 ;
                     02E0  1199 ;  set id into gbpb and link at end of the ifab's gbpb list
                     02E0  1200 ;
                     02E0  1201
08 A1   15   90      02E0  1202          MOVB    #GBPB$C_BID,GBPB$B_BID(R1)
44 BA   61   0E      02E4  1203          INSQUE  (R1),@IFB$L_BDB_BLNK(R10)
             05      02E8  1204 10$:     RSB                             ; note: r0 still has status code.
```

```
                     02E9  1206          .SBTTL  RMSRETBLB - BLB DEALLOCATION ROUTINE
                     02E9  1207 ;++
                     02E9  1208 ;
                     02E9  1209 ; RMSRETBLB - return specified BLB
                     02E9  1210 ;
                     02E9  1211 ; This routine deallocate the space used by a BLB, and removes it
                     02E9  1212 ; from the ifab list.
                     02E9  1213 ;
                     02E9  1214 ; Calling sequence:
                     02E9  1215 ;
                     02E9  1216 ;       BSBW    RMSRETBLB
                     02E9  1217 ;
                     02E9  1218 ; Input parameters:
                     02E9  1219 ;
                     02E9  1220 ;       R4 - address of BLB to be returned.
                     02E9  1221 ;       R10 - ifab address
                     02E9  1222 ;
                     02E9  1223 ; Output parameters:
                     02E9  1224 ;
                     02E9  1225 ;       R0 - R5 destroyed.
                     02E9  1226 ;
                     02E9  1227 ; Completion status:
                     02E9  1228 ;
                     02E9  1229 ;       none - success is assumed
                     02E9  1230 ;
                     02E9  1231 ;--
                     02E9  1232
                     02E9  1233 RMSRETBLB::
54   64   0F        02E9  1234          REMQUE  (R4),R4                 ; Remove from BLB chain.
  24 A4   D5        02EC  1235          TSTL    BLB$L_LOCK_ID(R4)       ; Make sure no lock is held.
     06   12        02EF  1236          BNEQ    10$                     ; NEQ lock not released.
53   5A   D0        02F1  1237          MOVL    R10, R3                 ; Free space header into R3.
   FEFB   31        02F4  1238          BRW     RMSRETBLK               ; Return the blb.
                     02F7  1239
                     02F7  1240 10$:     RMSPBUG FTL$_LOCKHELD           ; This is a problem.
```

```
                     02FE  1242          .SBTTL  RM$RETGBPB - GBPB DEALLOCATION ROUTINE
                     02FE  1243 ;++
                     02FE  1244 ;
                     02FE  1245 ; RM$RETGBPB - return specified GBPB
                     02FE  1246 ;
                     02FE  1247 ; This routine deallocate the space used by a GBPB, and removes it
                     02FE  1248 ; from the ifab list.
                     02FE  1249 ;
                     02FE  1250 ; Calling sequence:
                     02FE  1251 ;
                     02FE  1252 ;       BSBW    RM$RETGBPB
                     02FE  1253 ;
                     02FE  1254 ; Input parameters:
                     02FE  1255 ;
                     02FE  1256 ;       R4 - address of GBPB to be returned.
                     02FE  1257 ;       R10 - ifab address
                     02FE  1258 ;
                     02FE  1259 ; Output parameters:
                     02FE  1260 ;
                     02FE  1261 ;       R0 - R5 destroyed.
                     02FE  1262 ;
                     02FE  1263 ; Completion status:
                     02FE  1264 ;
                     02FE  1265 ;       none - success is assumed
                     02FE  1266 ;
                     02FE  1267 ;--
                     02FE  1268
                     02FE  1269 RM$RETGBPB::
54   64   0F        02FE  1270          REMQUE  (R4),R4                 ; Remove from BDB chain.
53   5A   D0        0301  1271          MOVL    R10, R3                 ; Free space header into R3.
     FEEB 31        0304  1272          BRW     RM$RETBLK               ; Return the gbpb.
```

```
                     0307  1274          .SBTTL  RM$RETBDB - BDB AND I/O BUFFER DEALLOCATION ROUTINE
                     0307  1275
                     0307  1276 ;++
                     0307  1277 ;
                     0307  1278 ; RM$RETBDB - return specified bdb
                     0307  1279 ;
                     0307  1280 ;  This routine deallocates the space occupied by a bdb,
                     0307  1281 ;  removes it from the ifab's bdb list, and deallocates
                     0307  1282 ;  the associated i/o buffer, if any.  Also adjust the
                     0307  1283 ;  buffer count if a buffer is deallocated.
                     0307  1284 ;
                     0307  1285 ;  The entry point RM$RETJNLBDB is used to deallocate a journal BDB and buffer.
                     0307  1286 ;  (Journaling specific BDBs and buffers are NOT linked into the IFAB BDB list.)
                     0307  1287 ;
                     0307  1288 ;  calling sequence:
                     0307  1289 ;
                     0307  1290 ;       BSBW    RM$RETBDB
                     0307  1291 ;
                     0307  1292 ;  input parameters:
                     0307  1293 ;
                     0307  1294 ;       r11     impure area address
                     0307  1295 ;       r10     ifab address
                     0307  1296 ;       r4      bdb address
                     0307  1297 ;
                     0307  1298 ;  implicit inputs:
                     0307  1299 ;
                     0307  1300 ;       none
                     0307  1301 ;
                     0307  1302 ;  output parameters:
                     0307  1303 ;
                     0307  1304 ;       r0 thru r5 destroyed
                     0307  1305 ;
                     0307  1306 ;  implicit outputs:
                     0307  1307 ;
                     0307  1308 ;       the free space and free page lists are updated.
                     0307  1309 ;
                     0307  1310 ;  completion codes:
                     0307  1311 ;
                     0307  1312 ;       none
                     0307  1313 ;
                     0307  1314 ;  side effects:
                     0307  1315 ;
                     0307  1316 ;       none
                     0307  1317 ;
                     0307  1318 ;--
                     0307  1319
                     0307  1320 RM$RETBDB::
0C   08 A4   91      0307  1321          CMPB    BDB$B_BID(R4),#BDB$C_BID ; is it a bdb?
        2F   12      030B  1322          BNEQ    ERRBUG1                  ; branch if not
                     030D  1323
                     030D  1324          ASSUME  BDB$L_FLINK     EQ      0
                     030D  1325
   54   64   0F      030D  1326          REMQUE  (R4),R4                  ; remove from ifab bdb list
                     0310  1327
                     0310  1328 ;
                     0310  1329 ; The next entry point id used to return journal BDBs and buffers.
                     0310  1330 ;
```

```
                     0310  1331
                     0310  1332 RMSRETJNLBDB::
                     0310  1333
                     0310  1334 ;
                     0310  1335 ;  return i/o buffer if any
                     0310  1336 ;
                     0310  1337
55   2C A4   3C     0310  1338          MOVZWL  BDB$W_ALLOC_SIZE(R4),R5 ; length of i/o buffer
        0C   13     0314  1339          BEQL    50$                     ; branch if none
                     0316  1340
                     0316  1341
        54   DD     0316  1342 20$:     PUSHL   R4                      ; save bdb addr
54   28 A4   D0     0318  1343          MOVL    BDB$L_ALLOC_ADDR(R4),R4 ; get buffer addr
      FE69   30     031C  1344          BSBW    RMSRETPAG               ; and deallocate the page(s)
        54 8ED0     031F  1345          POPL    R4                      ; restore bdb addr
                     0322  1346
                     0322  1347 ;
                     0322  1348 ;  check for bdb referenced in curbdb field of any irab and if so zero
                     0322  1349 ;
                     0322  1350
                     0322  1351          ASSUME  IFB$L_IRAB_LNK  EQ      IRB$L_IRAB_LNK
                     0322  1352
   50   5A   D0     0322  1353 50$:     MOVL    R10,R0                  ; get ifab addr to right reg
50   1C A0   D0     0325  1354 60$:     MOVL    IRB$L_IRAB_LNK(R0),R0   ; pick up next irab
        08   13     0329  1355          BEQL    70$                     ; branch if no more
54   20 A0   D1     032B  1356          CMPL    IRB$L_CURBDB(R0),R4     ; using this bdb?
        F4   12     032F  1357          BNEQ    60$                     ; branch if not
      20 A0   D4     0331  1358          CLRL    IRB$L_CURBDB(R0)        ; invalidate
        EF   11     0334  1359          BRB     60$                     ; and continue
                     0336  1360
                     0336  1361 ;
                     0336  1362 ;  now return the bdb
                     0336  1363 ;
                     0336  1364
53   5A   D0     0336  1365 70$:     MOVL    R10,R3                  ; copy of ifab addr
                     0339  1366                                      ; (free space header in this
                     0339  1367                                      ; page)
    FEB6   31     0339  1368          BRW     RMSRETBLK               ; return the bdb space
                     033C  1369
                     033C  1370 ;
                     033C  1371 ;  bad problem  -  the returning block was not a bdb!
                     033C  1372 ;
                     033C  1373
                     033C  1374 ERRBUG1:
                     033C  1375          RMSTBUG FTL$_BADBDB
```

```
                     0343  1377          .SBTTL  RMSALBLB - ALLOCATE BUCKET LOCK BLOCK
                     0343  1378 ;++
                     0343  1379 ;
                     0343  1380 ;       RMSALBLB
                     0343  1381 ;
                     0343  1382 ;       Function Description
                     0343  1383 ;
                     0343  1384 ;       Allocate and initialize static fields in the BLB.
                     0343  1385 ;       Link into BLB queue off the ifab.
                     0343  1386 ;
                     0343  1387 ;       Input Parameters:
                     0343  1388 ;
                     0343  1389 ;       R11     impure pointer
                     0343  1390 ;       R10     ifab address
                     0343  1391 ;
                     0343  1392 ;       Output Parameters:
                     0343  1393 ;
                     0343  1394 ;       R1      address of BLB
                     0343  1395 ;       R0      status code
                     0343  1396 ;
                     0343  1397 ;       Implicit outputs:
                     0343  1398 ;
                     0343  1399 ;       Many fields in the argument block portion of the BLB are initialized.
                     0343  1400 ;
                     0343  1401 ;       Completion codes:
                     0343  1402 ;
                     0343  1403 ;       Standard RMS - usually SUC or DME
                     0343  1404 ;
                     0343  1405 ;--
                     0343  1406
                     0343  1407 RMSALBLB::
          1C   BB    0343  1408          PUSHR   #^M<R2,R3,R4>                   ; Save registers
     51   5A   D0    0345  1409          MOVL    R10, R1                         ; Get free space header into R1.
     52   0E   D0    0348  1410          MOVL    #BLB$C_BLN/4, R2                ; Want length in longwords in R2.
        FE18   30    034B  1411          BSBW    RM$GETBLK                       ; Get the block.
     12 50   E9      034E  1412          BLBC    R0, 10$                         ; Exit on error.
08 A1   10   90      0351  1413          MOVB    #BLB$C_BID, BLB$B_BID(R1)       ; Set BID.
18 A1   04   D0      0355  1414          MOVL    #4, BLB$L_RESDSC(R1)            ; Stuff size of resource to 4.
     14 A1   DE      0359  1415          MOVAL   BLB$L_VBN(R1),-                 ; Set address of resource name
     1C A1           035C  1416                  BLB$L_RESDSC+4(R1)              ; into descriptor.
009C DA   61   0E    035E  1417          INSQUE  (R1), @IFB$L_BLBBLNK(R10)       ; Link into BLB queue.
          1C   BA    0363  1418 10$:     POPR    #^M<R2,R3,R4>                   ; restore registers
               05    0365  1419          RSB
```

```
                     0366  1421          .SBTTL  RMSALDJNLBUF - JOURNAL BDB AND I/O BUFFER ALLOCATION
                     0366  1422 ;++
                     0366  1423 ;
                     0366  1424 ; RMSALDJNLBUF - allocate buffer and bdb to go with it
                     0366  1425 ;
                     0366  1426 ;  this routine performs the following functions:
                     0366  1427 ;
                     0366  1428 ;       1. allocate and init a bdb
                     0366  1429 ;       2. allocate a journaling buffer (of an integral number of pages)
                     0366  1430 ;          NOTE: Journal BDBs ALWAYS have buffers.
                     0366  1431 ;
                     0366  1432 ;  calling sequence:
                     0366  1433 ;
                     0366  1434 ;       BSBW    RMSALDJNLBUF
                     0366  1435 ;
                     0366  1436 ;  input parameters:
                     0366  1437 ;
                     0366  1438 ;       r11     impure area address
                     0366  1439 ;       r10     ifab address
                     0366  1440 ;       r5      length of buffer in bytes
                     0366  1441 ;
                     0366  1442 ;  implicit inputs:
                     0366  1443 ;
                     0366  1444 ;       none
                     0366  1445 ;
                     0366  1446 ;  output parameters:
                     0366  1447 ;
                     0366  1448 ;       r4      address of bdb
                     0366  1449 ;       r3      address of buffer
                     0366  1450 ;       r2      total size in bytes of allocation
                     0366  1451 ;       r0      status code
                     0366  1452 ;       r1      destroyed
                     0366  1453 ;       r4,r5   also destroyed if r0 indicates an error
                     0366  1454 ;
                     0366  1455 ;  implicit outputs:
                     0366  1456 ;
                     0366  1457 ;  the affected free space and free page lists are updated.
                     0366  1458 ;  the BDB is NOT linked into the IFAB BDB list
                     0366  1459 ;
                     0366  1460 ;  completion codes:
                     0366  1461 ;
                     0366  1462 ;       standard rms, in particular, success or dme.
                     0366  1463 ;
                     0366  1464 ;  side effects:
                     0366  1465 ;
                     0366  1466 ;       none.
                     0366  1467 ;
                     0366  1468 ;--
                     0366  1469
                     0366  1470 RMSALDJNLBUF::
                     0366  1471
    27   10          0366  1472          BSBB    RMSALJNLBDB             ; get a journal bdb
  21 50  E9          0368  1473          BLBC    R0,60$                  ; branch on error
54   51  D0          036B  1474          MOVL    R1,R4                   ; save bdb addr
52   55  D0          036E  1475          MOVL    R5,R2                   ; move buffer len to right reg
    06   13          0371  1476          BEQL    20$                     ; eql then wants only bdb (no buffer)
  FC91   30          0373  1477          BSBW    RMSGETPAG               ; and get an i/o buffer
```

```
                     0376  1478                                    ; (len/addr returned in r2,r3)
          11 50   E9 0376  1479            BLBC    R0,50$          ; branch on error
                     0379  1480
                     0379  1481 ;
                     0379  1482 ; set buffer size and address into bdb
                     0379  1483 ;
                     0379  1484
    16 A4    55   B0 0379  1485 20$:      MOVW    R5,BDB$W_SIZE(R4)
    2C A4    55   B0 037D  1486           MOVW    R5,BDB$W_ALLOC_SIZE(R4)
    18 A4    53   D0 0381  1487           MOVL    R3,BDB$L_ADDR(R4)
    28 A4    53   D0 0385  1488           MOVL    R3,BDB$L_ALLOC_ADDR(R4)
             05   0389  1489           RSB                             ; note: r0 still valid.
                     038A  1490
                     038A  1491 ;
                     038A  1492 ;  error allocating the page  -  must return the bdb
                     038A  1493 ;
          84   10 038A  1494 50$:      BSBB    RM$RETJNLBDB            ; deallocate journal BDB
                     038C  1495
        FD5D   31 038C  1496 60$:      BRW     ERRDME                  ; restore error code
```

```
                     038F  1498          .SUBTITLE RMSALJNLBDB - JOURNAL BDB ALLOCATION
                     038F  1499
                     038F  1500 ;++
                     038F  1501 ;
                     038F  1502 ; RMSALJNLBDB -  allocate and initialize a journal BDB
                     038F  1503 ;
                     038F  1504 ;  calling sequence:
                     038F  1505 ;
                     038F  1506 ;       BSBW    RMSALJNLBDB
                     038F  1507 ;
                     038F  1508 ;  input parameters:
                     038F  1509 ;
                     038F  1510 ;       r11     impure area address
                     038F  1511 ;       r10     ifab address
                     038F  1512 ;
                     038F  1513 ;  implicit inputs:
                     038F  1514 ;
                     038F  1515 ;       none
                     038F  1516 ;
                     038F  1517 ;  output parameters:
                     038F  1518 ;
                     038F  1519 ;       r1      address of bdb
                     038F  1520 ;       r0      status code
                     038F  1521 ;       r2,r3,r4 destroyed
                     038F  1522 ;
                     038F  1523 ;  implicit outputs:
                     038F  1524 ;
                     038F  1525 ;  the bdb has its block length and block id fields filled in
                     038F  1526 ;
                     038F  1527 ;  completion codes:
                     038F  1528 ;
                     038F  1529 ;       standard rms, in particular, success and dme.
                     038F  1530 ;
                     038F  1531 ;  side effects:
                     038F  1532 ;
                     038F  1533 ;       none
                     038F  1534 ;
                     038F  1535 ;--
                     038F  1536
                     038F  1537 RMSALJNLBDB::
                     038F  1538
          51   5A    D0  038F  1539          MOVL    R10,R1                  ; copy ifab addr as this
                     0392  1540                                              ; defines the page for the
                     0392  1541                                              ; free space list header
          52   14    D0  0392  1542          MOVL    #BDB$C_BLN/4,R2         ; # longwords required
             FDCE    30  0395  1543          BSBW    RMSGETBLK               ; allocate zeroed space
                     0398  1544                                              ; (r1 set to addr)
            04 50    E9  0398  1545          BLBC    R0,10$                  ; branch on error
                     039B  1546 ;
                     039B  1547 ; set id into bdb
                     039B  1548 ;
      08 A1    0C    90  039B  1549          MOVB    #BDB$C_BID,BDB$B_BID(R1)
                     05  039F  1550 10$:     RSB                             ; note: r0 still has status code.
                     03A0  1551
                     03A0  1552          .END
```

```
$$.PSECT_EP                     = 00000000
$$ARGS                          = 00000002
$$RMSTEST                       = 0000001A
$$RMS_PBUGCHK                   = 00000010
$$RMS_TBUGCHK                   = 00000008
$$RMS_UMODE                     = 00000004
$$T1                            = 00000000
ASB$K_BLN_FAB                   = 00000160
ASB$K_BLN_FIX                   = 00000030
ASB$K_BLN_IDX                   = 00000200
ASB$K_BLN_REL                   = 000000C0
ASB$K_BLN_SEQ                   = 000000BC
BDB$B_BID                       = 00000008
BDB$C_BID                       = 0000000C
BDB$C_BLN                       = 00000050
BDB$K_BLN                       = 00000050
BDB$L_ADDR                      = 00000018
BDB$L_ALLOC_ADDR                = 00000028
BDB$L_FLINK                     = 00000000
BDB$W_ALLOC_SIZE                = 0000002C
BDB$W_SIZE                      = 00000016
BLB$B_BID                       = 00000008
BLB$C_BID                       = 00000010
BLB$C_BLN                       = 00000038
BLB$K_BLN                       = 00000038
BLB$L_LOCK_ID                   = 00000024
BLB$L_RESDSC                    = 00000018
BLB$L_VBN                       = 00000014
C511                              00000000 R      01
C512                              000001DE R      01
CMKRNL$_ARGLST                  = 00000008
CMKRNL$_NARGS                   = 00000002
CMKRNL$_ROUTIN                  = 00000004
CNTREG                            000000CD R      01
ERRBUG                            000000F2 R      01
ERRBUG1                           0000033C R      01
ERRDME                            000000EC R      01
ERRDME1                           000000EA R      01
ERRDME2                           000000E5 R      01
ERRDME_BR                         000002BC R      01
EXPREGERR                         000000B2 R      01
FTL$_BADBDB                     = FFFFFFFA
FTL$_LOCKHELD                   = FFFFFFD9
FTL$_SETPRTFAIL                 = FFFFFFFF
FWA$K_BLN                       = 0000093C
FWA$K_BLN_BUF                   = 0000093C
FWA$K_BLN_FWA                   = 000001F4
GBD$K_BLN                       = 00000028
GBH$K_BLN                       = 00000058
GBPB$B_BID                      = 00000008
GBPB$C_BID                      = 00000015
GBPB$C_BLN                      = 00000028
GBPB$K_BLN                      = 00000028
GBSB$K_BLN                      = 00000044
IDX$K_FIXED_BLN                 = 0000002C
IFB$B_BID                       = 00000008
IFB$C_BID                       = 0000000B
IFB$K_BLN_IDX                   = 000000B8
IFB$K_BLN_REL                   = 000000B4
IFB$K_BLN_SEQ                   = 000000AC
IFB$L_BDB_BLNK                  = 00000044
IFB$L_BLB_BLNK                  = 0000009C
IFB$L_IRAB_LNK                  = 0000001C
IMP$B_PROT                      = 00000002
IMP$L_FREEPGLH                  = 0000000C
IMP$L_IOSEGADDR                 = 00000004
IMP$L_IOSEGLEN                  = 00000008
IMP$V_IIOS                      = 00000000
IMP$V_NOPOBUFS                  = 00000005
IMP$W_RMSSTATUS                 = 00000000
IRB$B_BID                       = 00000008
IRB$C_BID                       = 0000000A
IRB$K_BLN_IDX                   = 000000C4
IRB$K_BLN_REL                   = 00000064
IRB$K_BLN_SEQ                   = 0000006C
IRB$L_CURBDB                    = 00000020
IRB$L_IFAB_LNK                  = 00000000
IRB$L_IRAB_LNK                  = 0000001C
MASK                            = 000001FF
MJB$K_BLN                       = 00000020
PSL$C_EXEC                      = 00000001
RJB$K_BLN                       = 0000000C
RLB$K_BLN                       = 0000001C
RM$ALBDB                          000002BF RG     01
RM$ALBLB                          00000343 RG     01
RM$ALDBUF                         00000296 RG     01
RM$ALDJNLBUF                      00000366 RG     01
RM$ALGBPB                         000002D4 RG     01
RM$ALJNLBDB                       0000038F RG     01
RM$BUG                            ******** RG   X 01
RM$GET1PAG                        00000004 RG     01
RM$GETBLK                         00000166 RG     01
RM$GETBLK1                        00000164 RG     01
RM$GETPAG                         00000007 RG     01
RM$GETSPC                         000000FC RG     01
RM$GETSPC1                        000000F9 RG     01
RM$GETSPC_ALT                     00000101 RG     01
RM$RET1PAG                        00000185 RG     01
RM$RETBDB                         00000307 RG     01
RM$RETBLB                         000002E9 RG     01
RM$RETBLK                         000001F2 RG     01
RM$RETBLK1                        000001F0 RG     01
RM$RETGBPB                        000002FE RG     01
RM$RETJNLBDB                      00000310 RG     01
RM$RETPAG                         00000188 RG     01
RM$RETSPC                         000001FA RG     01
RM$RETSPC1                        000001E2 RG     01
RMS$_DME                        = 000184D4
SETHDR1                           0000017A R      01
SETHDR3                           000001E5 R      01
SETPRT                            00000084 R      01
SFSB$K_BLN                      = 00000044
SLB$K_BLN                       = 00000018
SWB$K_BLN                       = 00000148
```

```
SYS$CNTREG                      ********  GX   01
SYS$EXPREG                      ********  GX   01
SYS$SETPRT                      ********  GX   01
```

Psect synopsis

```
PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
. ABS .                         00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
RM$RMS0                         000003A0  (  928.)  01 (  1.)    PIC   USR   CON   REL   GBL NOSHR   EXE   RD  NOWRT NOVEC BYTE
$ABS$                           00000000  (    0.)  02 (  2.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
```

Performance indicators

```
Phase                     Page faults     CPU Time        Elapsed Time
-----                     -----------     --------        ------------
Initialization                     36     00:00:00.07     00:00:00.92
Command processing                130     00:00:00.73     00:00:05.04
Pass 1                            509     00:00:20.82     00:00:54.27
Symbol table sort                   0     00:00:02.86     00:00:05.11
Pass 2                            249     00:00:05.05     00:00:14.31
Symbol table output                14     00:00:00.16     00:00:00.62
Psect synopsis output               2     00:00:00.02     00:00:00.02
Cross-reference output              0     00:00:00.00     00:00:00.00
Assembler run totals              942     00:00:29.72     00:01:20.30

The working set limit was 1950 pages.
114923 bytes (225 pages) of virtual memory were used to buffer the intermediate code.
There were 100 pages of symbol table space allocated to hold 1894 non-local and 48 local symbols.
1552 source lines were read in Pass 1, producing 16 object records in Pass 2.
51 pages of virtual memory were used to define 50 macros.
```

Macro library statistics

```
Macro library name                                Macros defined
------------------                                --------------
_$255$DUA28:[RMS.OBJ]RMS.MLB;1                              26
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                               6
_$255$DUA28:[SYSLIB]STARLET.MLB;2                           14
TOTALS (all libraries)                                      46

2143 GETS were required to define 46 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:RMOBUFMGR/OBJ=OBJ$:RMOBUFMGR MSRC$:RMOBUFMGR/UPDATE=(ENH$:RMOBUFMGR)+EXECML$/LIB+LIB$:RMS/LIB
```